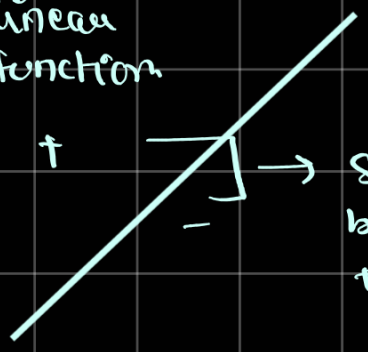
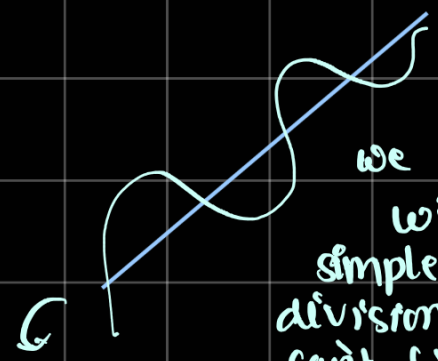
Linear function



→ simple binary +1- or 0/1 division

non-linear curve

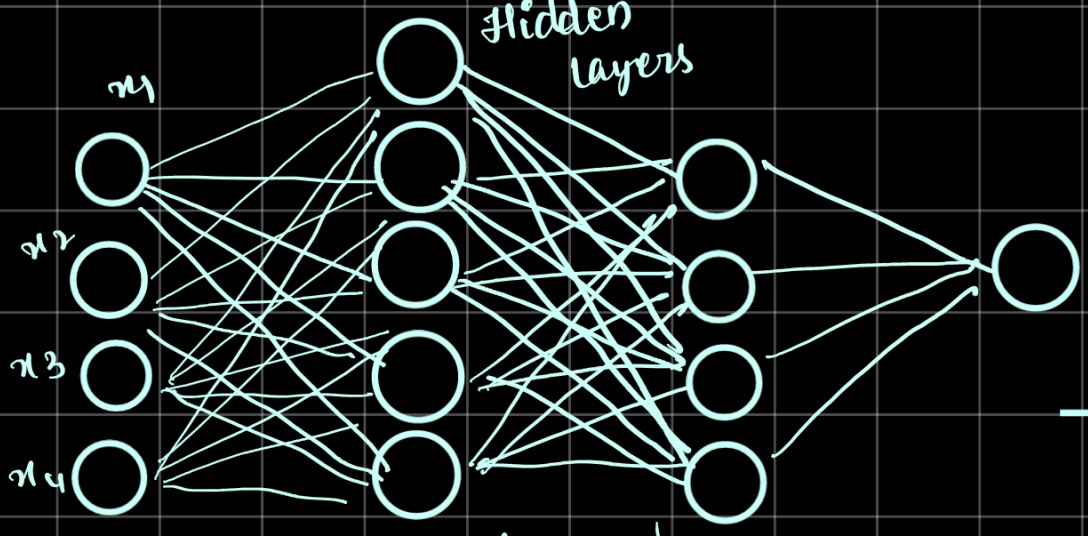


we can't go with a simple binary division, we can't find correlation

thus we find individual relations

ie $x_1 \rightarrow y_1$, $x_2 \rightarrow y_2$, ⋮, $x_k \rightarrow y_k$ → finding a correlation → then fitting it with with a function

feature engineering → domain knowledge → for a machine to predict this features → we use deep learning

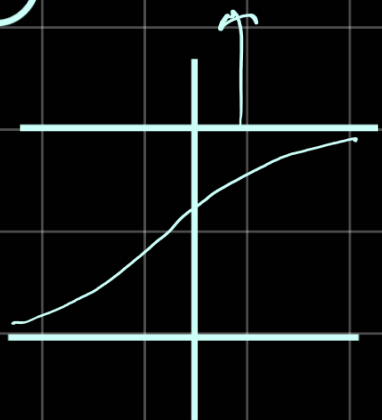


logistic regression

$\theta_1 x_1 + \theta_2 x_2 + \cdots \theta_k x_k$

$\sigma(\theta_1 x_1 + \theta_2 x_2 + \cdots \theta_k x_k)$

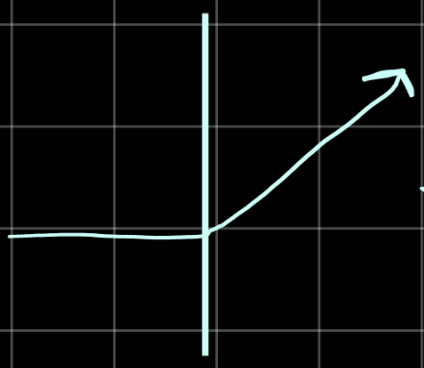
adds non-linearity



σ sigmoid graph

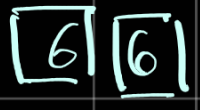
Alternative activation function to sigmoid function is

relu
= ↳ (Popular)

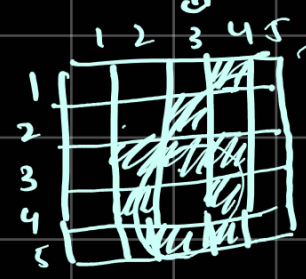


→ partially linear function
→ Discontinuous but Lasso applicable.

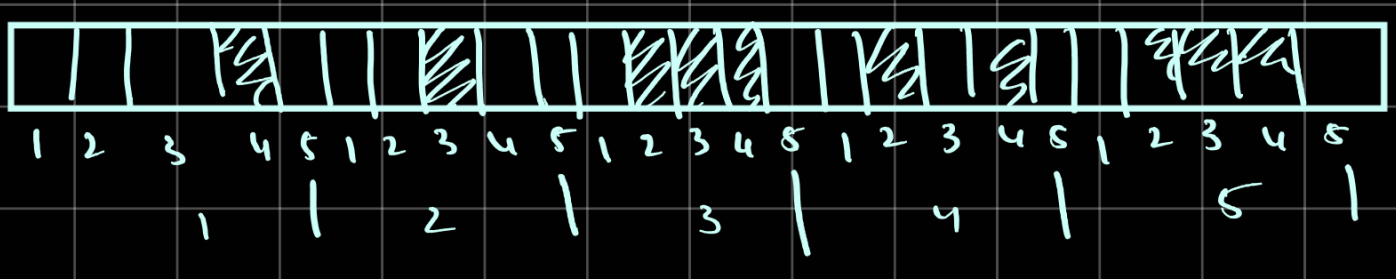
↳ captures features from local to global to determine the universal features of an element in a neural network.

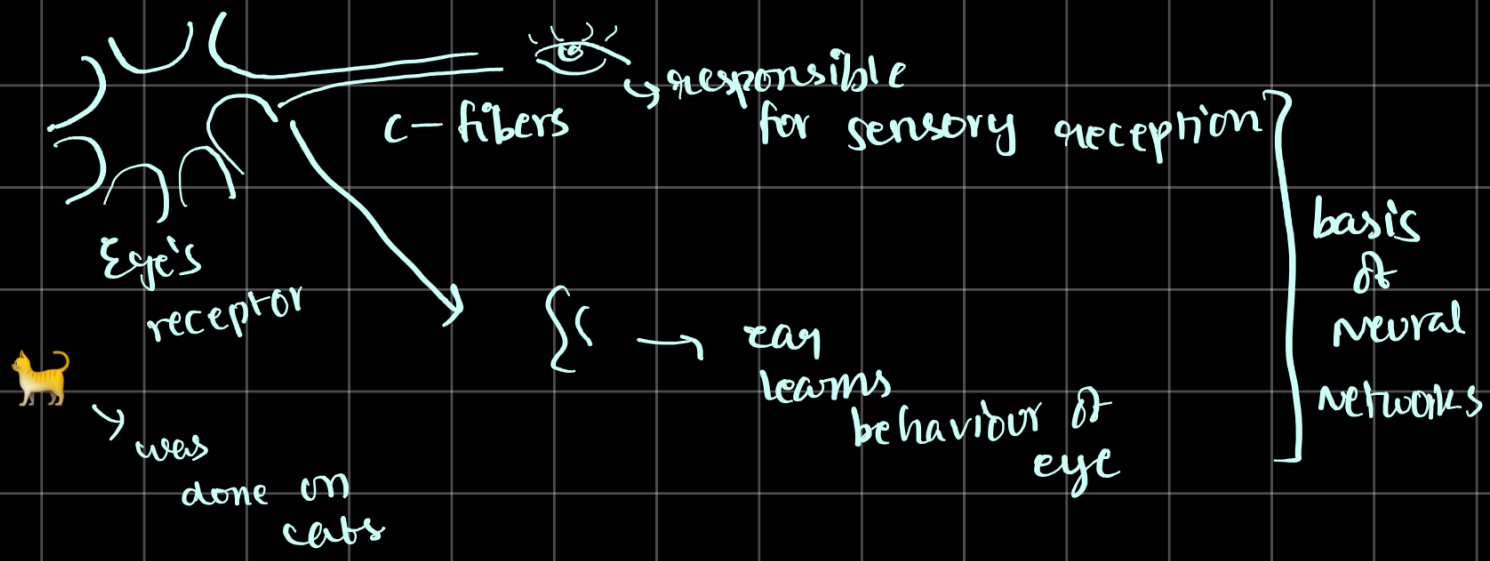


sending an image in form of pixels



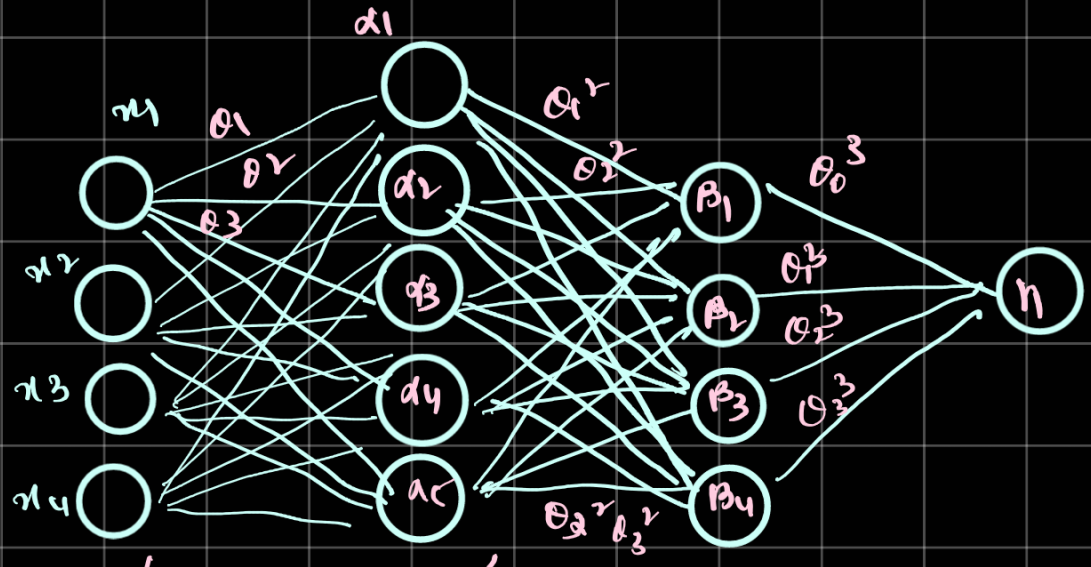
↓
as array
↓





→ if m → inputs
n → hidden networks
↳ m × n

n → inputs
p → hidden networks
↳ n × p



$\theta_{01}\ \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 \cdots \theta_k x_4$

↓

1st layer ; input

$\theta_0 + \theta_1^2 a_1 + \theta_2^2 a_2 + \theta_3^2 a_2 \cdots \theta_k^2 a_k$

Hidden layer 1

$\theta_0 + \theta_1^3 a_2 + \theta_2^3 a_2 + \theta_3^3 a_2 + \cdots \theta_k^3 a_k$

Hidden layer k.

Loss function J

$$J = -Y\ln(Y') - (1-Y)\ln(1-Y')$$

$$\frac{\partial J}{\partial \theta_1^2} = \frac{\partial J}{\partial B_1} \times \frac{\partial B_1}{\partial \theta_{12}}$$ → using chain rule

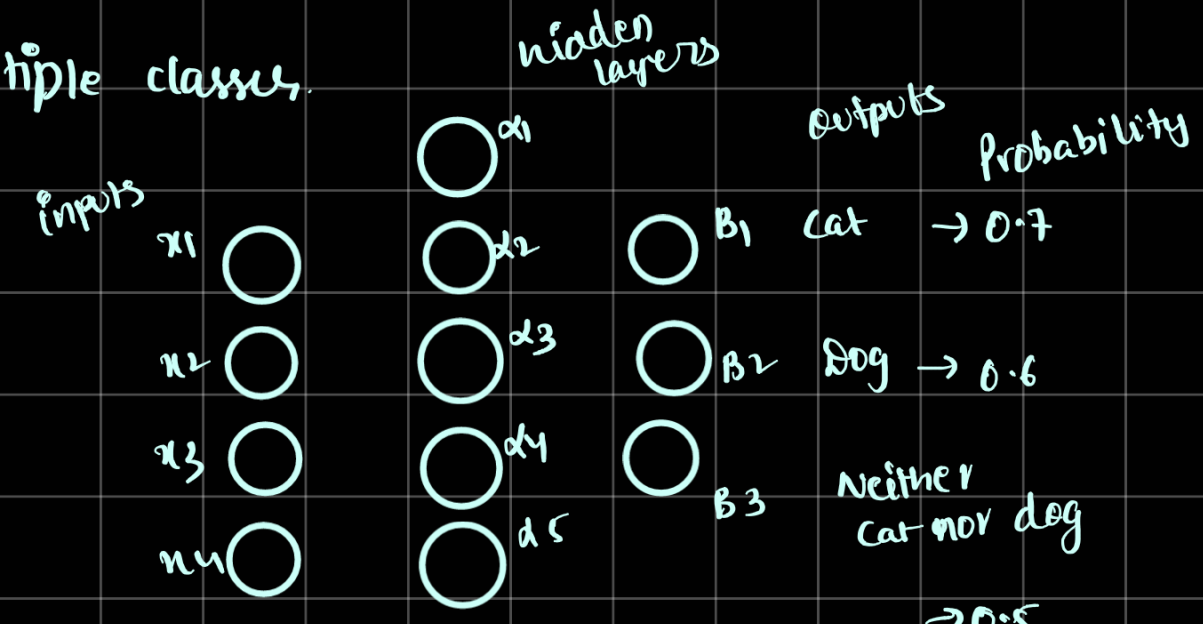
↓

each hidden unit focuses on a local feature ; one class at a time

→ forward passes in neural networks act as 'tensors' not as loops

;
$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 \\ x_1^{23} & x_2^3 & x_3^3 & x_4^3 \end{bmatrix}_{3\times 4} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}_{4\times 1}$$ → Simple tensor multiplication.

→ Multiple classes.



Soft max

↳ when probabilities are really close, it scales the differences so that the differences become significant

Say probability of cat = 0.7

$$\text{Soft max} = \frac{e^{0.7}}{e^{0.7}e^{0.6} + e^{0.5}}$$

(0.6 ↓ dog; 0.5 ↓ neither cat | dog)

(Linear Layer)

ReLU → Rectified Linear Unit → non activation function

↳ The output of the 1st hidden layer is sent to a ReLU function to apply the function

↓

$$\text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Label is the one which has the highest probability

→ Depth of network is n-1 cause input is not counted as a layer.

Forward computation & Backward loss propagation
↓
rates get adjusted
↳ the 0's are the rates.

Hyperparameters :- The specific parameters that we give while training our model.

→ cross entropy loss

Schocastic gradient descent
↓
has an optimiser
↓
creates back propagation.
↑
upgrades weights

step-1 → sends images
↓
predicts loss
↓
then propagated back
↓
SGD works

2 →

3 →

4 →

🙃

Accuracy ↑es and loss ↓es for each epoch

Too many epoch → over fitting

constant value → model converged.

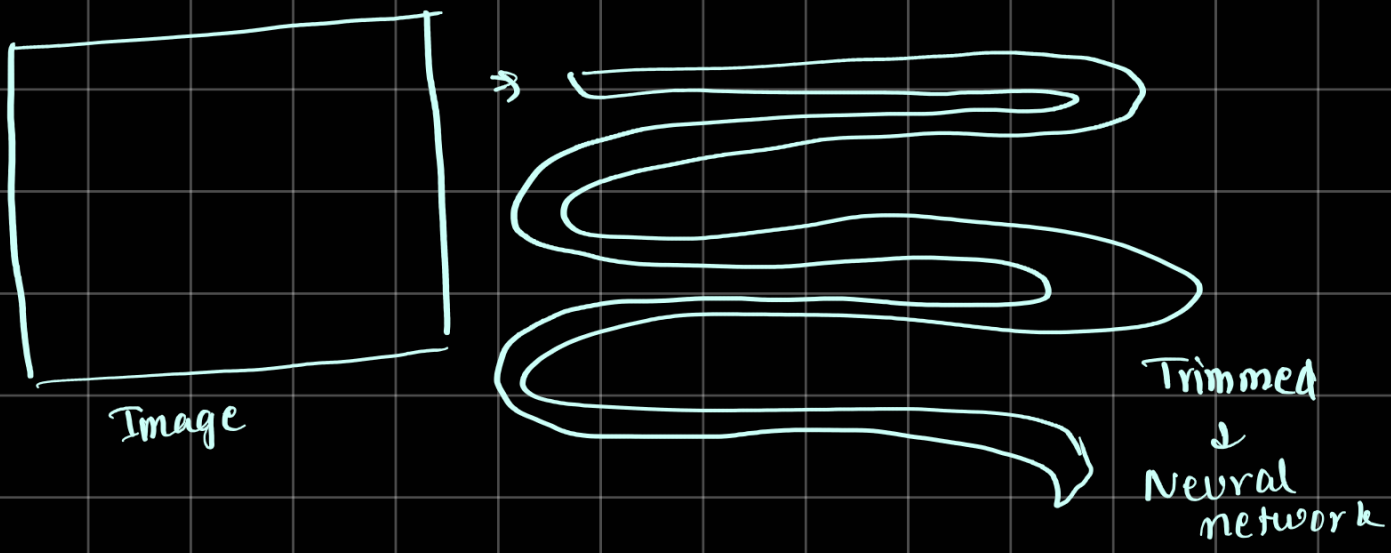
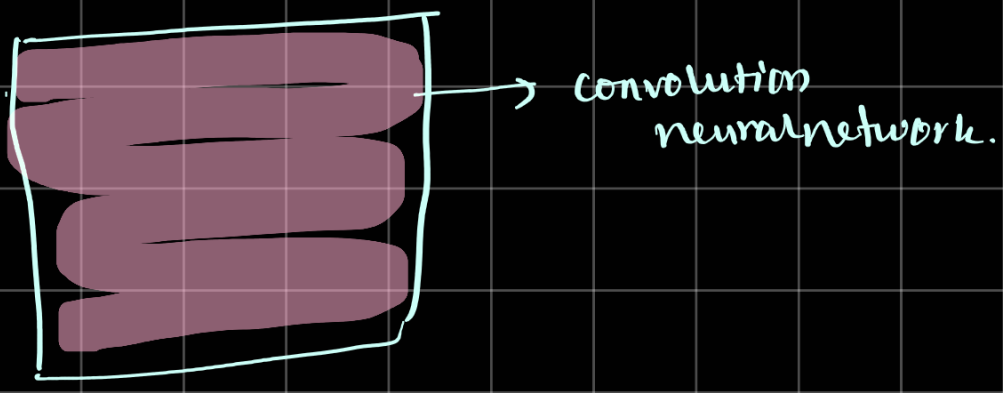


Image
Trimmed
&
Neural
network


convolution
neural network.